AN INTERACTIVE SYSTEM FOR THE DEFINITION OF A SEMANTIC DATA MODEL

by

GREGORY DALE WOOD

B.S., Kansas State University, 1974

A MASTERS THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

Approved by:

Major Professor

Table of Contents

List of Figures

Chapter 1
Introduction

The conceptual modeling phase of database design, has been a time consuming manual task, with inconsistent results. One tool for this task, the Semantic Data Model (SDM), was developed by Hammer and McLeod (3,4,5) to describe the semantics of application environments with a formal syntax, while retaining an understandable relationship to the actual data. The development of a strict syntax for an SDM, and an interactive system for its entry, maintenance and verification will, hopefully, simplify conceptual modeling and produce more reliable results. The additional step of the automated generation of a static Data Dictionary (DD) from the SDM should also simplify the transition to logical database design.

The syntax developed by Lane (8) from the work of Hammer and McLeod (3,4,5), is based on the grouping of identifiable units, or entities, whether concrete or abstract, into classes. Three types of classes are defined; Base Classes, Sub Classes and Grouping Classes. Base Classes can be defined independently from other classes, while Sub and Grouping Classes must be defined in relation to a parent class. This relationship is based on attributes of the classes or their members.

An interactive system for the entry, maintenance and verification of an SDM is a specific example of interactive computerized application systems. Such systems use a two way dialog between the user and the computer, or more accurately the software executing on the computer, to perform a specific task or set of tasks. Much work has been done to

optimize the design of such systems and the productivity of their users. Andriole (1) identified six phases in the design of such systems.

The first phase, system targeting, should identify the prospective user, his background, experience level, current working environment, as well as requirements of the proposed system.

Phase two, system modeling, should provide a more detailed list of the functions to be included in the system, and their interrelations. As Gains (2) observed, it is also important to understand the users current method of accomplishing the task or tasks, so as to present him with a recognizable dialog. The term "user friendly" has been widely used to describe this objective, but perhaps a more complete description was given by Gains (2), when he said such a system should "present an understandable and sympathetic face to its user".

The third phase, software design, is most critical, but will most likely not succeed unless it is based on an accurate completion of the first two phases. Also, the impact of Andrioles (1) phase four, hardware selection, on software design, can at times be large enough to justify the reversal of these two phases.

Dialog design, the first task of software design in an interactive system, should be tailored to the users experience level. Those users with little experience require a system with less flexibility, that is, a small number of ways to accomplish each task, and less complexity, or a small number of options at any specific time. Inexperienced users are also more comfortable with a computer initiated dialog, although as Gains (2) pointed out, the user should dominate the dialog, so as to avoid user uncertainty and the resulting dissatisfaction. Computer

initiated dialogs can take the form of question and answer sessions, menus, fill in the blank forms, or some combination of these, although it is important to maintain a consistent use of form across similar functions.

At the other end of the spectrum are user initiated systems for the more experienced user. These can be much more flexible and complex, and can be implemented by techniques such as command or natural languages. Mixtures of the two types of systems are possible, such as using a menu for a simple subtask, and a command language for a more complex subtask, but as Monk (7) advised, care must be taken to avoid confusing the user.

Display design also requires consistency, specifically in the use of emphasizing techniques such as highlighting, flashing, multicolor, or reverse video. Placement of information on the display is also of interest. Monk (7) recommended that the upper right quadrant of the display be used for high priority information, and that no more than twenty five percent of the display be filled, while Miller (6) suggested that seven or less options for a unidimensional variable be presented at one time.

This report describes the application of these principles to the design of an interactive system for the creation of a correct SDM. The primary target population of this system is made up of undergraduate students in database design classes. Chapter 2 gives a more detailed description of the problem and Chapter 3 continues with the system design. Chapter 4 describes the system implementation, and Chapter 5 gives a summary and suggestions for future directions.

Chapter 2

Problem description

An SDM can be viewed as having only two major types of components, classes and attributes, which an interactive system must address. Each has a variety of required and optional sub-components.

Class definition will require a class name and optionally a class description, then will diverge depending on which type of class is being defined, Base, Sub or Grouping.

Base Classes will require an indication of the acceptability of multiple members with identical attribute values, and optionally, one or more identifiers, where an identifier is an attribute or combination of two attributes, the value of which will specify a member, or multiple members if duplicates are allowed.

Sub-Classes will require the class name of the parent class and an indication of the type of relationship between the two. There are five types of relationships between Sub-Classes and their parent classes; specification, set operator defined, format, attribute predicate and existence.

The simplest of these is specification, in which the relationship is declared to exist. An example of this would be the relationship of core courses in the Computer Science Masters Program to all courses in Computer Science Masters Program. Although some criteria, other than arbitrary selection, probably exists in this case, as in most cases of Sub-Class by specification, those criteria are esoteric enough to defy easy definition by any of the other methods available.

A more easily defined and probably the most common, relationship is

created by the set operators; union, intersection and difference. Using these operators, the membership of a Sub-Class is based on membership of two other classes, or occasionally, one other class and the universal set, such as the definition of Graduate Computer Science Courses as the intersection of Graduate Courses with Computer Science Courses.

The format relationship is primarily used to subset the Base Class STRINGS into a more specific Sub-Class. For example the COURSE ID Sub-Class could be given the format of a two character alphabetic code followed by a three digit number.

Attribute predicate relationships are based on the existence of a logical relationship between the value of a specific attribute of the members of the parent class, and either a literal or the value of a second attribute. For example, GRADUATE COURSES can be defined as those for which the value of the attribute Course# of the parent class COURSES is greater than 600.

The last type, existence, is used by Hammer and McLeod (5) to show a relationship similar to intersection, but membership is based on the value of an attribute in one of the classes rather than membership in the classes. In their example, DANGEROUS_CAPTAINS is defined as the subclass of OFFICERS which also exist as values of the attribute Involved_captain of the class INCIDENTS. This relationship is easily, and more understandably, stated as an attribute predicate, the subset of the class OFFICERS, whose value for the attribute Name, is equal to the value of the attribute Involved Captain of the INCIDENTS class. It would seem that the existence relationship can always be stated as an attribute predicate, and thus could be excluded from this system.

The final type of class, Grouping Classes, has a very basic difference from Base and Sub-Classes, a clear understanding of which is necessary for their correct use. Base and Sub classes have, as their members, individual items or entities, whereas in a Grouping Class each member is a group of items or entities, each of which must be a member the parent class. The definition of a Grouping Class requires the class name of the parent class, as well as one of two grouping techniques.

The first such technique, grouping by common value of some member attribute, will create groups with the value of the indicated attribute as member names. An example would be the Grouping Class DEPARTMENTAL COURSES, or the grouping, from the COURSES class, of the attribute Courses# on the common value of the attribute Department, which would have as members, CS, EE, etc., each of which would consist of a group of courses.

The second method of grouping, enumeration, simply consists of the listing of two or more Sub-Classes, such as the grouping of CS COURSES and MATH COURSES into TECHNICAL COURSES, the contributing classes becoming members of the Grouping Class.

Hammer and McLeod (5) give a third method of grouping, user controlled or specification, and use as an example the CONVOYS class, whose members are user specified groups of ships. Although CONVOYS is a good example of a Grouping Class, to arrive at it by specification is ungainly. It would be preferable to include a member attribute called Present Convoy in the Base Class SHIPS, on which a common value grouping could be made. It seems likely that any case of grouping by means of user specification could be accomplished by one of the previous methods,

possibly after the addition of an attribute to the parent class, or the definition of additional Sub-Classes. The ease of these additions, in an interactive system, argues against the inclusion of specification grouping.

In addition to class definition, two types of attributes can be defined. Class attributes are features of the class as a whole, such as the Total Active attribute of the FACULTY class. Member attributes are features of individual members of the class. Both types of attributes require a name and a value class, where a value class refers to a class whose members are the potential values for this attribute. They will also require an indication of the number of concurrent values possible for the attribute, as well as the acceptability of null or changeable values. For member attributes, the existence of an inverse relationship with another attribute, in this or another class, may be noted. For example, the Instructor attribute of the COURSES class is the inverse of the Courses Taught attribute of the FACULTY class.

Member attributes will also require an indication of whether the attribute values must exhaust the value class, and the acceptability of duplication of any value of the attribute from member to another. Optionally each attribute may have a description and a method of value derivation. Both class and member attributes may use statistical derivation. This includes the functions minimum, maximum, average, sum, total number or number of unique, as applied to another attribute in the class. A simple example would be a member attribute of Grade Point in the class of STUDENTS, which could be defined as the average of the values of attribute Course Grade. Member attributes can be derived in

several other ways.

Their values can be obtained from members of this or another class having common values for a specified attribute, using the value of yet another attribute from each such member to generate a value, statistical if the derived attribute is to be single valued, or a set if it is to be multivalued. For example, the Hours Taught attribute of the FACULTY class can be obtained by summing the values for the attribute Credit for members of the COURSES class having a common value for the attribute Instructor. The multivalued attribute Courses Taught could be derived similarly, although without the summation.

Hammer and McLoed (3,4,5) frequently refer to mappings in their syntax, defining them recursively as an attribute name or an attribute name dot mapping. They explain the concept thus defined, as a direct reference to the value of an attribute of an attribute. A flaw in this explanation is immediately apparent, since attributes do not have attributes. Their first example, Captain dot Name, is completely useless as it derives its value from itself. If it were changed to be somewhat more useful, say Captain dot Pay, the actual method of derivation would be to take the value of the attribute Pay, from the member of the OFFICERS class whose value for the attribute Name is equal to the value of the attribute Captain of the current class. Obviously the syntax given does not supply enough information to resolve this derivation. One possible syntax to rectify this situation would be:

```
MAPPING <-
 [ATTRIBUTE_NAME onto ATTRIBUTE_NAME of CLASS_NAME using ATTRIBUTE_NAME;
  MAPPING        onto ATTRIBUTE_NAME of CLASS_NAME using ATTRIBUTE_NAME]
```

The first attribute is defined to belong to the current class. This syntax contains enough information, although the recursive case may introduce an unacceptable amount of complexity into an interactive system. An examination of the need for a recursive case will show that it can safely be omitted from this system.

The three level example given by Hammer and McLeod (5), Captain dot Superiors dot Name, displays the same problem encountered with the first example, Captain dot Name. The value class for superiors is most likely already NAMES, and thus the mapping should be reduced to Captain dot Superiors. If the value class for superiors happened to be SERIAL#, as in "name, rank and", then a true recursion would be required to derive the desired value. First a set of serial numbers would be obtained by mapping Captain onto Name of OFFICERS using Superiors. Then the actual value would be derived by mapping that result onto Serial# of OFFICERS using Name. Rather than use the recursive syntax, an intermediate attribute, Captains Superiors, could be defined and used in the mapping for Captains Superiors Name.

Ranking, by increasing or decreasing order, of the value of some other member attribute, within the class or within a group of members sharing a common value of yet another attribute, can be used to create a value. The Seniority attribute of the FACULTY class, for example, is an ordering of the class on the value of the attribute Years of Service.

A boolean value of true or false can be derived from the inclusion or exclusion, of the member in question, in another class. For example, the Required attribute of the class CS COURSES, is true if the course is a member of class CS COURSES REQUIRED.

One of the more complex derivations, recursion, can generate a large set of values for a multivalued attribute through repeated applications of an attribute. For example the attribute Children can be repeatedly applied to generate the multivalued attribute Descendants. Hammer and McLeod (5) suggest the ability to limit the number of levels of recursion.

Another complex derivation is the subsetting of another multivalued attribute, based on the satisfaction of an attribute predicate similar to that used in the specification of Sub-Classes. For example, the attribute Morning Sessions of the COURSES class, consists of the subset of Meeting Times with values between 0600 and 1200.

Still other sets of values can be derived by the application of the set operators union, intersection and difference, to other multivalued attributes, such as the definition of Nonmorning Sessions as the difference between Meeting Times and Morning Sessions.

The final derivation uses mathematical computation on one or more other attributes and zero or more literals. An example would be the attribute Session Length, which could be calculated from Meeting Time and Ending Time.

Chapter 3

System design

The interactive SDM to DD system is intended as a database design tool for use by undergraduate computer science students and professional database designers. Both groups of users should be familiar with the selected hardware, IBM compatible personal computers, as well as MS/DOS based software. The professional users should have a working knowledge of SDMs, whereas the students will have only classroom exposure.

This experience level would lead to the use of a computer initiated dialog, if the number of functions needed is not prohibitively large. From the users point of view, entry and maintenance of an SDM can be seen as one function, only varying in the original contents of the SDM in question. Parsing of a complete SDM is another required function and generation of a DD from an SDM is a third. A fourth function, which may not be apparent from the problem definition, is the deletion of an entire SDM. This seems to be a sufficient set of functions to satisfy the requirements of the system, with the possible addition of an exit option, and is a reasonable number of options for a menu driven system.

The first function, entry and maintenance of the SDM, includes a large number of sub-functions. Most of these involve the definition of components of an SDM and are best implemented with the fill in the blank format. This format provides the user with guidance in the completion of the definition, and is less time consuming than a question and answer session. A few of the sub-functions have enough options to support the further use of the menu format. The remainder of this chapter details the displays used in the system.

Access to the system is by entering the system name, SDM. The user is then presented with the Primary Selection Menu (figure 3.1). As on all screens in the system, entry fields are displayed in reverse video and are preceded by an arrow and a short item of text identifying the intended contents of the field.

```
SDM ************************** MENU ********************************


            Option (D = Delete S D M
                    E = Edit S D M
                    G = Generate DATA DICTIONARY from S D M
                    P = Parse S D M
                    X = Exit) ==>



*****************************************************************************
 F1=Help     F3=         F5=         F7=         F9=         ESC=MS/DOS
 F2=         F4=         F6=         F8=         F10=        C/R=Enter
*****************************************************************************
```

Figure 3.1: Primary Selection Menu

If help is selected (F1) before any option is selected, or at any time when no error has been detected, a full screen description of the currently active piece of the system and any related SDM syntax, is displayed. If a character other than those letters shown as options, is entered, an error message is flashed in the upper right hand corner of the screen (figure 3.2). If help is selected at this point, or whenever a flashing error message is being displayed, a full screen description of that error is provided.

```
SDM ************************** MENU ****************   INVALID OPTION


            Option (D = Delete S D M
                    E = Edit S D M
                    G = Generate DATA DICTIONARY from S D M
                    P = Parse S D M
                    X = Exit) ==>


*******************************************************************************
 F1=Help     F3=         F5=         F7=         F9=         ESC=MS/DOS
 F2=         F4=         F6=         F8=         F10=        C/R=Enter
*******************************************************************************
```

Figure 3.2: Flashing error message

Once an option, other than X, has been selected, a screen for the identification of the desired SDM is displayed (figure 3.3). Each SDM is given a full path name, including the drive on which it exists, and the automatically added extension SDM. The SDM definition will then be stored on disk with this path name. Any valid MS/DOS path name can be used, up to the limit of three directories and a file name.

Figure 3.3: SDM Identification screen

Movement in the system is controlled by the function keys. From any SDM entry screen, as opposed to the menu, id or help screens, the function keys institute the following actions.

F1 - Help

F2 - Display the current class

F3 - Display the first class attribute of the current class

F4 - Display the first member attribute of the current class

F5 - Insert an additional component of type currently displayed

F6 - Delete the component currently displayed

F7 - Display the previous component of type currently displayed

F8 - Display the next component of type currently displayed

F9 - Update the component currently displayed

F10 - End

The ESC key will affect a nondestructive exit to MS/DOS.

If the previously identified SDM already exists, its first class definition will be displayed. If not, the Class Definition screen (figure 3.4) will be displayed with blank entry fields.

Name and type are required fields and their omission will cause a flashing error message. Additional screens are used to complete the class definition, depending on the type selected.

```
SDM ***************************** CLASS ********************************


        Name ==>

        Type (B = Base, S = Sub, G = Grouping) ==>

        Description ==>



*****************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.4: Class Definition screen

If type Base was chosen, the Base Class screen (figure 3.5) is displayed. The validity of members with duplicate identifiers, in this class, must be indicated, as well as the makeup of the identifiers, if any exist. To allow the entry of multiple identifiers for the class, the identifier fields will be blanked and redisplayed each time the enter key is pressed, until the update function key is pressed. Each identifier is comprised of from one to five attributes.

```
SDM ************************** Base Class ****************************

    Duplicates allowed (Y, N) ==>

    Identifier: attribute     ==>
                + attribute ==>
                + attribute ==>
                + attribute ==>
                + attribute ==>

    Identifier: attribute     ==>
                + attribute ==>
                + attribute ==>
                + attribute ==>
                + attribute ==>

******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
******************************************************************************
```

Figure 3.5: Base Class definition screen

If type Sub is chosen one or two additional screens are needed to complete the definition. The first (figure 3.6) requires the name of the parent class and a choice of class relationship. The second depends upon the type of relationship chosen. No second screen is needed for specification.

```
SDM ************************** Sub-Class ******************************


        Parent Class ==>

        Relation type (1 = specification,
                       2 = set operator
                       3 = format
                       4 = attribute predicate) ==>


********************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
********************************************************************************
```

Figure 3.6: Sub-Class definition screen

If the set operator relationship is chosen, the second screen (figure 3.7) requires the type of set operation to be used and the names of the two Sub-Classes involved.

```
SDM ********************* Sub-Class Set Operator ************************


    Operation (U = union, D = difference, I = intersection) ==>

                  of Sub-Class ==>

                 and Sub-Class ==>



*****************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.7: Sub-Class Set Operator screen

If format is chosen, the second screen (figure 3.8) requires a size range, and optionally a value range for each position in the size range. The maximum size available in this system is 17.

```
SDM ********************** Sub-Class Format **************************

Size from ==>    to ==>    01 from ==>   to ==>
                           02 from ==>   to ==>
                           03 from ==>   to ==>
                           04 from ==>   to ==>
                           05 from ==>   to ==>
                           06 from ==>   to ==>
                           07 from ==>   to ==>
                           08 from ==>   to ==>
                           09 from ==>   to ==>
                           10 from ==>   to ==>
                           11 from ==>   to ==>
                           12 from ==>   to ==>
                           13 from ==>   to ==>
                           14 from ==>   to ==>
                           15 from ==>   to ==>
                           16 from ==>   to ==>
                           17 from ==>   to ==>
*******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*******************************************************************************
```

Figure 3.8: Sub-Class Format screen

The second screen (figure 3.9) for the last and most complex type of relationship, attribute predicate, requires the name of the first attribute, the predicate operator and either a literal or the name of a second attribute. Of the ten operators, the first six are familiar and should be easy to remember. The last four are used to indicate some set relationship. The first two of which are only valid if the first of the two attributes is multivalued, and the last two of which are only valid if the second attribute is multivalued.

```
SDM ****************** Sub-Class Attribute Predicate *******************

            Attribute ==>

                 (GT = greater then,
                  LT = less then,
                  EQ = equal to,
                  NE = not equal to,
                  GE = greater then or equal to,
                  LE = less then or equal to,
                  CT = contains,
                  PC = properly contains,
                  IN = is contained in,
                  PI = is properly contained in) ==>

            Attribute or 'literal' ==>

*****************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.9: Sub-Class Attribute Predicate screen

If Grouping Class was chosen the Grouping Class Definition screen (figure 3.10) is displayed. It requires the name of the parent class and either the name of an attribute to be used for grouping by common value, or a list of Sub-Classes to be grouped.

```
SDM ************************ Grouping Class ***************************

          Parent class ==>

          on like value of attribute ==>

          or of Sub-Classes ==>
                            ==>
                            ==>
                            ==>
                            ==>
                            ==>
                            ==>
                            ==>
                            ==>
                            ==>

*******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*******************************************************************************
```

Figure 3.10: Grouping Class Definition screen

The Class Attribute Definition screen (figure 3.11) requires only the attribute name, but will validate any of the optional fields given.

```
SDM *************************** Class Attribute ***********************

   Name ==>                      Value class ==>

   Description ==>

   Non-null  (Y, N) ==>

   Changeable (Y, N) ==>

   Number of values from ==>    to ==>

   Derivation (MIN, MAX, AVG, SUM, TOT, UNQ) ==>

      of attribute ==>

*****************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.11: Class Attribute Definition screen

Member attribute definition uses one or more screens, depending on the options chosen. The first (figure 3.12) is similar to that used for class attribute definition, with the additional options of Inverse, Exhaustive and Overlap, and the simplification of the Derivation choice to Y or N.

```
SDM ************************** Member Attribute ***********************

   Name ==>                      Value class ==>

   Description ==>


   Inverse of attribute ==>
              of class ==>
   Non-null    (Y, N) ==>
   Changeable  (Y, N) ==>
   Exhaustive  (Y, N) ==>
   Overlap     (Y, N) ==>
   Number of values from ==>    to ==>
   Derivation (Y, N) ==>
      or Match on attribute ==>
                    of class ==>
            using attribute ==>

*******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*******************************************************************************
```

Figure 3.12: Member Attribute Definition screen

If an attribute derivation is indicated, a menu for selection of derivation type (figure 3.13) is displayed. Each choice from this menu involves an additional screen with which to describe the derivation.

```
SDM ******************** Member Attribute Derivation Type **************

            Derivation (1 = Mapping,
                        2 = Ordering,
                        3 = Recursion,
                        4 = Set operation,
                        5 = Statistical,
                        6 = Attribute predicate,
                        7 = Mathematical) ==>

********************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
********************************************************************************
```

Figure 3.13: Member Attribute Derivation Type screen

The first option, mapping (figure 3.14), requires identification of the attribute name from the current class, to be used in the mapping, the attribute name and class name onto which the mapping is to be done and the attribute name from that class from which the derived value is to be taken.

```
SDM ************ Member Attribute Derivation - Mapping ***************


    Mapping of attribute ==>

           onto attribute ==>                    of class ==>

          using attribute ==>



*****************************************************************************
 F1=Help      F3=C Attr   F5=Insert   F7=Previous F9=Update    ESC=MS/DOS
 F2=Class     F4=M Attr   F6=Delete   F8=Next     F10=End      C/R=Enter
*****************************************************************************
```

Figure 3.14: Member Attribute Derivation - Mapping screen

Ordering (figure 3.15) requires the name of the attribute upon which the ordering is to be based and an indication of the direction, ascending or descending, in which to evaluate the ordering. Optionally a second attribute can be named to limit the domain of the ordering. In effect, this is a two level sort. The class is first sorted on the second attribute named and then on the first, in the order specified. The value derived is then the relative position within common values of the second attribute.

```
SDM ************* Member Attribute Derivation - Ordering ***************


      Ordering of attribute ==>

      in (A = ascending, D = descending) ==>   order

      within attribute ==>



*****************************************************************************
 F1=Help      F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class     F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.15: Member Attribute Derivation - Ordering screen

The recursion derivation (figure 3.16) requires only the name of the attribute to be recursively applied. Optionally a limit to the number of levels of recursion can be given.

```
SDM ************ Member Attribute Derivation - Recursion **************



                On attribute ==>

                up to ==>   levels




********************************************************************************
 F1=Help      F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class     F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
********************************************************************************
```

Figure 3.16: Member Attribute Derivation - Recursion screen

Derivation based on set operations (figure 3.17) requires the identification of the operation used, and the names of the attributes involved.

```
SDM ********** Member Attribute Derivation - Set ******************


          (U = union, D = difference, I = intersection) ==>

           of attribute ==>

          and attribute ==>



*****************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*****************************************************************************
```

Figure 3.17: Member Attribute Derivation - Set screen

Similarly statistical derivation (figure 3.18) requires only the type of analysis to be used and the name of the attribute to be used.

```
SDM ********** Member Attribute Derivation - Statistical ***************

                          (MIN = minimum,
                           MAX = maximum,
                           AVG = average,
                           SUM = sum,
                           TOT = total number,
                           UNQ = number of unique) ==>

                           of attribute ==>

*******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*******************************************************************************
```

Figure 3.18: Member Attribute Derivation - Statistical screen

Mathematical derivation (figure 3.19) requires the name of at least one attribute and one or more combinations of an operator and a literal or an attribute name.

```
SDM ******** Member Attribute Derivation - Mathematical ***************

                attribute ==>
          (+, -, *, /) ==>  attribute or 'literal' ==>
          (+, -, *, /) ==>  attribute or 'literal' ==>
          (+, -, *, /) ==>  attribute or 'literal' ==>
          (+, -, *, /) ==>  attribute or 'literal' ==>
          (+, -, *, /) ==>  attribute or 'literal' ==>


******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
******************************************************************************
```

Figure 3.19: Member Attribute Derivation - Mathematical screen

The final type of derivation, attribute predicate (figure 3.20), requires one attribute name, the identification of a predicate and either a literal or the name of another attribute.

```
SDM ********** Member Attribute Derivation - Predicate ***************

            Attribute ==>

                 (GT = greater then,
                  LT = less then,
                  EQ = equal to,
                  NE = not equal to,
                  GE = greater then or equal to,
                  LE = less then or equal to,
                  CT = contains,
                  PC = properly contains,
                  IN = is contained in,
                  PI = is properly contained in) ==>

            Attribute or 'literal' ==>

*******************************************************************************
 F1=Help     F3=C Attr   F5=Insert   F7=Previous F9=Update   ESC=MS/DOS
 F2=Class    F4=M Attr   F6=Delete   F8=Next     F10=End     C/R=Enter
*******************************************************************************
```

Figure 3.20: Member Attribute Derivation - Predicate screen

Chapter 4

System Implementation

The hardware chosen to support this system, IBM compatible personal computers, imposes some limitations on the choice of language used for the implementation. In order to support a full screen system, with displays such as those shown in the previous chapter and software controlled function keys and cursor movement, the language chosen must be able to directly access the video display and the keyboard. It also should handle complex data structures and direct access files to allow access to individual components of the SDM. Of the languages available on the personal computer, C is well suited to these requirements.

Several display and keyboard input and output functions are supported. From this base, a more powerful set of functions (see appendix A) was developed.

Using these functions, a general purpose display and keyboard control routine (see appendix B) was written for use in the interactive system (see appendix E). This routine accepts as parameters:

1) an index into an array of screen definitions;

2) the field number at which to position the cursor;

3) the address of a buffer in which to store any input;

4) an index into an array of error messages (see appendix D).

Shown in figure 4.1 is the Primary Selection Menu screen as defined for display by this routine. All of the screen definitions are given in appendix C. Any input fields defined are displayed in reverse video and primed with any data in the buffer at the time of the call.

The keyboard is also controlled by this routine. Any non-control character is written to the buffer and echoed to the screen. Control characters are translated to meaningful values and returned to the caller. Cursor movement is calculated against the known input fields and cannot reach any other position of the screen, thus insuring that any data entered directly corresponds to a known entry field.

```
 1,                  /* number of input fields    */
00,                  /* original cursor field     */
00,                  /* previous screen (none)    */
{                    /* length, row, col          */
{  1, 14, 39}},
{                    /* screen text lines         */
"SDM ******************************** MENU ***************************",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                 Option (D = Delete S D M                             ",
"                                                                      ",
"                         E = Edit S D M                               ",
"                                                                      ",
"                         G = Generate DATA DICTIONARY from S D M      ",
"                                                                      ",
"                         P = Parse S D M                              ",
"                                                                      ",
"                         X = Exit) ==>                                ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"**********************************************************************",
" F1=Help    F3=        F5=         F7=         F9=        ESC=MS/DOS ",
" F2=        F4=        F6=         F8=         F10=       C/R=Enter  ",
"**********************************************************************"
}
```

Figure 4.1: Primary Selection Menu screen definition

Storage of an SDM is by fixed length records, each of which is directly accessible by its record number. Each record contains a single class or attribute definition. The structure of an entire SDM is realized by storing pointers in the form of record numbers, in each component record. This chain of pointers originates in a header record which contains the record number of the first class in the SDM. Each class record, whether base or non-base, has pointers to the next and the previous class record. Any pointer which has no current object has a value of binary zero. Class records also have pointers to the first class attribute and the first member attribute associated with that class. Each of these attribute records contain pointers to the next and the previous attribute of the same type. Rather than store Pointers to the associated class in each attribute record, a single system variable saves the record number of the last class record displayed.

Traversing these chains is fairly straight forward. The function key F2 selects the record indicated by the system variable for previous class. Similarly F3 selects the first class attribute from that class, and F4 selects the first member attribute. F7, previous record of the current type, uses the backward pointer stored in each record, and F8, next record of current type, uses the corresponding forward pointer. Any request encountering a zero forward pointer, that is for a non-existent next record is translated to an insert.

The insertion and deletion of records is slightly more complex than the simple traversing of pointer chains, since it is here that the chains must be created and maintained. Any scheme to make reasonable use of storage space, must include the ability to reuse space occupied

by deleted records. To accomplish this, the header record contains a pointer to the last deleted record. This pointer is maintained in a system variable and the header record is updated only when the SDM file is closed. When a record is deleted, the records indicated by its next and previous pointers are updated to reflect its deletion. If the deleted record happens to be the first class attribute or the first member attribute of a class, its previous record pointer will be zero. In this case the class record, indicated by the system variable for last class displayed, is updated. The system variable for last deleted record is stored into the next record pointer of the record being deleted and is then updated with the record number of that record. This creates a chain of pointers to deleted records available for reuse.

This process is essentially reversed when inserting a new record. First the system variable for last deleted record is checked. If it is zero the new record is simply appended to the end of the file. If, however, the last deleted record is not zero, the value of the forward pointer from the record indicated is used to update the last deleted record system variable, and the record is used to store the new SDM component. The new record is then inserted into the pointer chain, its forward and backward pointers set to indicate the records between which it is being inserted, and their corresponding pointers set to indicate the new record.

Validation of the SDM definition, although not yet implemented, will be on two levels. First the contents of each field entered will be checked for a valid character set for that particular field, and after an entire definition is entered, it will be passed to parsing routine.

Chapter 5

Summary

The design and implementation of an interactive system for the creation of a correct SDM involved three main tasks:

1) analysis of the SDM syntax to find a minimum functional subset;
2) creation of an easily understandable format for its entry;
3) writing code to handle the display of and the input to these formats, and the storage of the SDM thus defined.

The first task resulted in three fairly significant changes to the syntax used by Hammer and Mcloed (3,4,5):

1) elimination of the existence relationship Sub-Group;
2) elimination of user specification grouping;
3) creation of a new syntax for Mapping.

Smaller changes include the limitation of references to Mappings to member attribute derivation, and the implementation of only the non-recursive form of the Mapping syntax.

The second task required the identification of the various sub-components of an SDM and a logical organization thereof. This resulted in the creation of 17 input screens to accept data for SDM definition.

The third task involved writing a set of display and keyboard control functions and a single general screen handling routine. It also required the design of a storage structure and linkages to allow the retrieval, insertion and deletion of SDM components.

In addition to the changes made to the syntax for the SDM, several areas remain worthy of further inspection.

Most obvious is the attribute predicate, whether used as a subclassing technique or for member attribute derivation. Its meaning is only clear when the second half of the predicate is a literal.

The recursive derivation, although its meaning seems clear, quite likely suffers from a lack of information similar to that discovered in mappings.

Problems of this sort will become critical when the syntax is used as input to an automated data dictionary generator. These small items of missing information, which the human reader fills in without notice, will create absolute blocks to the translation to a data dictionary language.

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
/*                CURSES for the AT&T 6300                 */
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
# define TRUE    (1)
# define FALSE   (0)
# define ERR     (-1)
# define OK      (0)
# define mvaddstr(y,x,str)    (move(y,x)==ERR?ERR:addstr(str))
# define mvaddch(y,x,ch )     (move(y,x)==ERR?ERR:putch(ch))
# define addch(ch)            putch(ch)
#include <stdio.h>
#include <conio.h>
#include <signal.h>
#include <dos.h>
#include <process.h>
#define LINES 24
#define COLS 80
#define VIDEO_INT 0x10
#define KEYBD_INT 0x16
#define KEY_ECHO  0x01
union REGS regs, outregs;
clear(){
   regs.h.ah = 6;      /*  scroll up function          */
   regs.h.al = 0;      /*  code to blank screen        */
   regs.h.ch = 0;      /*  upper left row              */
   regs.h.cl = 0;      /*  upper left column           */
   regs.h.dl = 80;     /*  lower right column          */
   regs.h.dh = 24;     /*  lower right row             */
   regs.h.bh = Al;     /*  attribute byte              */
   int86(VIDEO_INT, &regs, &regs);
   return;
}
move(R,C)int R,C;{
   regs.h.ah = 2;      /* position cursor function  */
   regs.h.dh = R;      /* row position of cursor    */
   regs.h.dl = C;      /* column position of cursor */
   regs.h.bh = 0;      /* current page              */
   int86(VIDEO_INT, &regs, &regs);
   return;
}
addstr(str)char *str;{
   while (*str) putch(*str++);
}
```

```
wattron(w,at) int *w, at; {
   regs.h.ah = 6;      /*  scroll up function          */
   regs.h.al = 0;      /*  code to blank screen        */
   regs.h.ch = w[0];   /*  upper left row              */
   regs.h.cl = w[1];   /*  upper left column           */
   regs.h.dl = w[2];   /*  lower right column          */
   regs.h.dh = w[3];   /*  lower right row             */
   regs.h.bh = at;     /*  attribute byte              */
   int86(VIDEO_INT, &regs, &regs);
   return;
}
scroll(ulr, ulc, lrr, lrc, attr, line) {
   regs.h.ah = 6;    /*  scroll up function          */
   regs.h.al = line; /*  # lines to scroll           */
   regs.h.ch = ulr;  /*  upper left row              */
   regs.h.cl = ulc;  /*  upper left column           */
   regs.h.dl = lrc;  /*  lower right column          */
   regs.h.dh = lrr;  /*  lower right row             */
   regs.h.bh = attr; /*  attribute byte              */
   int86(VIDEO_INT, &regs, &regs);
   return;
}
getch() {
   regs.h.ah = 0;
   int86(KEYBD_INT, &regs, &regs);
   return(regs.h.al);
}
newwin(nr,nc,sr,sc) int nr, nc, sr, sc;{
   if (nr == 0 ) nr = LINES - sr;
   if (nc == 0 ) nc = COLS  - sc;
   return;
}
initscr(){
}
endwin(){
}
cbreak(){
}
noecho(){
}
refresh(){
}
```

Appendix B

General screen control routine

```
scio(n, c, b, e) int n, c, e; char b[];{
   int i, j, k, m, l, ch, w[4];
   clear();                                          /* clear the screen  */
   for (i=0;i<24;i++)
      mvaddstr(i,0,scr[n].text[i]);                  /* display tesxt     */
   for (i=k=0;i<scr[n].fields;i++) {
      w[0] = w[3] = scr[n].floc[i][1];               /* set input fields  */
      w[1] = scr[n].floc[i][2];                      /* to reverse video  */
      w[2] = w[1] + scr[n].floc[i][0] - 1;
      wattron(w,RV);
      move(scr[n].floc[i][1],scr[n].floc[i][2]); /* and initialize    */
      for (j=0;j<scr[n].floc[i][0];j++)              /* with values from  */
         addch(b[k++]);                              /* buffer            */
   }
   if (e) {                                          /* if an error is    */
      w[0] = w[3] = 0;                               /* indicated, set    */
      w[1] = 57;                                     /* error field to    */
      w[2] = 76;                                     /* reverse video and */
      wattron(w,RV);                                 /* display the error */
      mvaddstr(0,57,msg[e-1]);                       /* message           */
   }
   for (i=j=0;i<c;i++) k += scr[n].floc[i][0];
   move(scr[n].floc[i][1],scr[n].floc[i][2]);
   refresh();
   while ((ch=getch())!=ENTER) {                     /* while the entered */
      for (k=m=0;m<i;m++)                            /* character is not  */
         k += scr[n].floc[m][0];                     /* ENTER, echo it if */
      if (!iscntrl(ch)) addch(b[k+j++]=ch);          /* not control char  */
      else switch(ch) {
         case 0 : switch (regs.h.ah) {
            case 59:                                                /*     f 1    */
            case 60:                                                /*     f 2    */
            case 61:                                                /*     f 3    */
            case 62:                                                /*     f 4    */
            case 63:                                                /*     f 5    */
            case 64:                                                /*     f 6    */
            case 65:                                                /*     f 7    */
            case 66:                                                /*     f 8    */
            case 67:                                                /*     f 9    */
            case 68: return(regs.h.ah);                             /*     f 10   */
            case 71: i=j=0; break;                                  /*  home      */
            case 72: if (--i < 0)                                   /*  up arrow  */
                        i = scr[n].fields-1;                        /* (back tab) */
                     j = 0; break;
```

```
            case 75: if (--j < 0) {                          /* right arrow*/
                        if (--i < 0)
                           i = scr[n].fields-1;
                        j = scr[n].floc[i][0]-1;
                     } break;
            case 77: if (++j > scr[n].floc[i][0]-1)          /* left arrow */
                        j = scr[n].floc[i][0]; break;
            case 79: j = scr[n].floc[i][0]-1; break;         /* e o field  */
            case 80: j = scr[n].floc[i][0]; break;           /* down arrow */
            default: addch(regs.h.ah);
          } break;
          case  8: if (--j < 0) {                            /* backspace  */
                        if (--i < 0)
                           i = scr[n].fields-1;
                        j = scr[n].floc[i][0]-1;
                     } break;
          case  9: j = scr[n].floc[i][0]; break;             /* tab        */
          case 15: if (--i < 0)                              /* back tab   */
                        i = scr[n].fields-1;
                   j = 0; break;
          case 27: esc();                                    /* escape     */
          default: addch(regs.h.ah);
       }
       if (j > scr[n].floc[i][0]-1) {              /* maintain cursor   */
          j = 0;                                   /* within defined    */
          if (++i > scr[n].fields-1)               /* input fields      */
             i = 0;
       }
       move(scr[n].floc[i][1],scr[n].floc[i][2]+j);
       refresh();
    }
    return(OK);
}
```

Appendix C

Screen Definitions

```
struct screens {
   int fields;                          /* number of input fields */
   int cursor;                          /* original cursor field  */
   int prevscr;                         /* previous screen number */
   int floc[40] [3];                    /* length, row, col       */
   char text [24] [80];                 /* screen text lines      */
} scr[] = {
{
 1,                  /* number of input fields    */
00,                  /* original cursor field     */
00,                  /* previous screen (none)    */
{                    /* length, row, col          */
{  1, 14, 39}},
{                    /* screen text lines         */
"SDM ******************************** MENU *************************",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                Option (D = Delete S D M                            ",
"                                                                    ",
"                        E = Edit S D M                              ",
"                                                                    ",
"                        G = Generate DATA DICTIONARY from S D M     ",
"                                                                    ",
"                        P = Parse S D M                             ",
"                                                                    ",
"                        X = Exit) ==>                               ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"**********************************************************************",
" F1=Help     F3=         F5=          F7=          F9=        ESC=MS/DOS ",
" F2=         F4=         F6=          F8=          F10=       C/R=Enter  ",
"**********************************************************************"
},
}
```

```
,{
 5,                  /* number of input fields    */
 0,                  /* original cursor field     */
 0,                  /* previous screen number    */
{                    /* length, row, col          */
    { 1,  7, 23},
    { 8, 10, 23},
    { 8, 10, 41},
    { 8, 10, 59},
    { 8, 13, 23}},
{                    /* screen text lines         */
"SDM ************************* IDENTIFICATION **********************",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"           Drive   ==>                                              ",
"                                                                    ",
"                                                                    ",
"           Path    ==>          Path ==>          Path ==>          ",
"                                                                    ",
"                                                                    ",
"           SDMname ==>                                              ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=         F5=          F7=          F9=        ESC=MS/DOS ",
" F2=        F4=         F6=          F8=          F10=       C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
 3,                  /* number of input fields    */
00,                  /* original cursor field     */
 0,                  /* previous screen number    */
{                    /* length, row, col          */
      {17,  7, 17},
      { 1, 10, 51},
      {44, 13, 24}},
{                    /* screen text lines         */
"SDM ****************************** CLASS ***************************",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"        Name ==>                                                     ",
"                                                                     ",
"                                                                     ",
"        Type (B = Base, S = Sub, G = Grouping) ==>                   ",
"                                                                     ",
"                                                                     ",
"        Description ==>                                              ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"*********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"*********************************************************************"
},
}
```

```
,{
11,                     /* number of input fields    */
00,                     /* original cursor field     */
 2,                     /* previous screen number    */
{                       /* length, row, col          */
      { 1,  3, 34},
      {17,  6, 34},
      {17,  7, 34},
      {17,  8, 34},
      {17,  9, 34},
      {17, 10, 34},
      {17, 13, 34},
      {17, 14, 34},
      {17, 15, 34},
      {17, 16, 34},
      {17, 17, 34}},
{                       /* screen text lines         */
"SDM ************************** Base Class ************************",
"                                                                  ",
"                                                                  ",
"    Duplicates allowed (Y, N) ==>                                 ",
"                                                                  ",
"                                                                  ",
"    Identifier: attribute     ==>                                 ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                                                                  ",
"                                                                  ",
"    Identifier: attribute     ==>                                 ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                + attribute ==>                                   ",
"                                                                  ",
"                                                                  ",
"******************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"******************************************************************"
},
}
```

```
,{
 2,                  /* number of input fields    */
00,                  /* original cursor field     */
 2,                  /* previous screen number    */
{                    /* length, row, col          */
      {17,  5, 27},
      { 1, 14, 54}},
{                    /* screen text lines         */
"SDM ************************** Sub Class ************************",
"                                                                 ",
"                                                                 ",
"                                                                 ",
"                                                                 ",
"          Parent Class ==>                                       ",
"                                                                 ",
"                                                                 ",
"          Relation type (1 = specification,                      ",
"                                                                 ",
"                         2 = set operator                        ",
"                                                                 ",
"                         3 = format                              ",
"                                                                 ",
"                         4 = attribute predicate) ==>            ",
"                                                                 ",
"                                                                 ",
"                                                                 ",
"                                                                 ",
"                                                                 ",
"*****************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"*****************************************************************"
},
}
```

```
,{
 3,                  /* number of input fields    */
00,                  /* original cursor field     */
 4,                  /* previous screen number    */
{                    /* length, row, col          */
      { 1,  7, 65},
      {17, 10, 36},
      {17, 13, 36}},
{                    /* screen text lines         */
"SDM ********************* Sub-Class Set Operator ******************",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"     Operation (U = union, D = difference, I = intersection) ==>    ",
"                                                                    ",
"                                                                    ",
"                   of Sub-Class ==>                                 ",
"                                                                    ",
"                                                                    ",
"                  and Sub-Class ==>                                 ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
36,                     /* number of input fields    */
00,                     /* original cursor field     */
 4,                     /* previous screen number    */
{                       /* length, row, col          */
      { 2,  2, 14},
      { 2,  2, 24},
      { 1,  2, 39},
      { 1,  2, 48},
      { 1,  3, 39},
      { 1,  3, 48},
      { 1,  4, 39},
      { 1,  4, 48},
      { 1,  5, 39},
      { 1,  5, 48},
      { 1,  6, 39},
      { 1,  6, 48},
      { 1,  7, 39},
      { 1,  7, 48},
      { 1,  8, 39},
      { 1,  8, 48},
      { 1,  9, 39},
      { 1,  9, 48},
      { 1, 10, 39},
      { 1, 10, 48},
      { 1, 11, 39},
      { 1, 11, 48},
      { 1, 12, 39},
      { 1, 12, 48},
      { 1, 13, 39},
      { 1, 13, 48},
      { 1, 14, 39},
      { 1, 14, 48},
      { 1, 15, 39},
      { 1, 15, 48},
      { 1, 16, 39},
      { 1, 16, 48},
      { 1, 17, 39},
      { 1, 17, 48},
      { 1, 18, 39},
      { 1, 18, 48}},
```

```
{                    /* screen text lines          */
"SDM ********************** Sub-Class Format ***********************",
"                                                                    ",
"Size from ==>    to ==>    01 from ==>   to ==>                     ",
"                           02 from ==>   to ==>                     ",
"                           03 from ==>   to ==>                     ",
"                           04 from ==>   to ==>                     ",
"                           05 from ==>   to ==>                     ",
"                           06 from ==>   to ==>                     ",
"                           07 from ==>   to ==>                     ",
"                           08 from ==>   to ==>                     ",
"                           09 from ==>   to ==>                     ",
"                           10 from ==>   to ==>                     ",
"                           11 from ==>   to ==>                     ",
"                           12 from ==>   to ==>                     ",
"                           13 from ==>   to ==>                     ",
"                           14 from ==>   to ==>                     ",
"                           15 from ==>   to ==>                     ",
"                           16 from ==>   to ==>                     ",
"                           17 from ==>   to ==>                     ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
 3,                  /* number of input fields    */
00,                  /* original cursor field     */
 4,                  /* previous screen number    */
{                    /* length, row, col          */
      {17,  3, 27},
      { 2, 14, 54},
      {17, 16, 40}},
{                    /* screen text lines         */
"SDM ***************** Sub-Class Attribute Predicate ***************",
"                                                                    ",
"                                                                    ",
"             Attribute ==>                                          ",
"                                                                    ",
"                  (GT = greater then,                               ",
"                   LT = less then,                                  ",
"                   EQ = equal to,                                   ",
"                   NE = not equal to,                               ",
"                   GE = greater then or equal to,                   ",
"                   LE = less then or equal to,                      ",
"                   CT = contains,                                   ",
"                   PC = properly contains,                          ",
"                   IN = is contained in,                            ",
"                   PI = is properly contained in) ==>               ",
"                                                                    ",
"             Attribute or 'literal' ==>                             ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
12,                     /* number of input fields    */
00,                     /* original cursor field     */
 2,                     /* previous screen number    */
{                       /* length, row, col          */
      {17,  4, 28},
      {17,  6, 42},
      {17,  8, 33},
      {17,  9, 33},
      {17, 10, 33},
      {17, 11, 33},
      {17, 12, 33},
      {17, 13, 33},
      {17, 14, 33},
      {17, 15, 33},
      {17, 16, 33},
      {17, 17, 33}},
{                       /* screen text lines         */
"SDM ************************* Grouping Class ************************",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"           Parent class ==>                                           ",
"                                                                      ",
"           on like value of attribute ==>                             ",
"                                                                      ",
"           or of Sub-Classes ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                             ==>                                      ",
"                                                                      ",
"                                                                      ",
"**********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"**********************************************************************"
},
}
```

```
,{
 9,                  /* number of input fields    */
00,                  /* original cursor field     */
 0,                  /* previous screen number    */
{                    /* length, row, col          */
      {17,  3, 12},
      {17,  3, 46},
      {44,  5, 19},
      { 1,  8, 24},
      { 1, 10, 24},
      { 2, 12, 29},
      { 2, 12, 39},
      { 3, 14, 49},
      {17, 16, 23}},
{                    /* screen text lines         */
"SDM ************************ Class Attribute ***********************",
"                                                                     ",
"                                                                     ",
"   Name ==>                  Value class ==>                         ",
"                                                                     ",
"   Description ==>                                                   ",
"                                                                     ",
"                                                                     ",
"   Non-null  (Y, N) ==>                                              ",
"                                                                     ",
"   Changeable (Y, N) ==>                                             ",
"                                                                     ",
"   Number of values from ==>    to ==>                               ",
"                                                                     ",
"   Derivation (MIN, MAX, AVG, SUM, TOT, UNQ) ==>                     ",
"                                                                     ",
"      of attribute ==>                                               ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"*********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"*********************************************************************"
},
}
```

```
,{
15,                     /* number of input fields    */
00,                     /* original cursor field     */
 0,                     /* previous screen number    */
{                       /* length, row, col          */
      {17,  2, 12},
      {17,  2, 46},
      {44,  4, 19},
      {17,  7, 28},
      {17,  8, 28},
      { 1,  9, 25},
      { 1, 10, 25},
      { 1, 11, 25},
      { 1, 12, 25},
      { 2, 13, 29},
      { 2, 13, 39},
      { 1, 14, 25},
      {17, 15, 32},
      {17, 16, 32},
      {17, 17, 32}},
{                       /* screen text lines         */
"SDM ********************** Member Attribute ***********************"
"                                                                   "
"   Name ==>                    Value class ==>                     "
"                                                                   "
"   Description ==>                                                 "
"                                                                   "
"                                                                   "
"   Inverse of attribute ==>                                        "
"              of class ==>                                         "
"   Non-null   (Y, N) ==>                                           "
"   Changeable (Y, N) ==>                                           "
"   Exhaustive (Y, N) ==>                                           "
"   Overlap    (Y, N) ==>                                           "
"   Number of values from ==>    to ==>                             "
"   Derivation (Y, N) ==>                                           "
"      or Match on attribute ==>                                    "
"                    of class ==>                                   "
"            using attribute ==>                                    "
"                                                                   "
"                                                                   "
"*******************************************************************"
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS "
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  "
"*******************************************************************"
},
}
```

```
,{
 1,                  /* number of input fields    */
 0,                  /* original cursor field     */
10,                  /* previous screen number    */
{                    /* length, row, col          */
{ 01, 16, 46})},
{                    /* screen text lines         */
"SDM ***************** Member Attribute Derivation Type **************",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"            Derivation (1 = Mapping,                                  ",
"                                                                      ",
"                        2 = Ordering,                                 ",
"                                                                      ",
"                        3 = Recursion,                                ",
"                                                                      ",
"                        4 = Set operation,                            ",
"                                                                      ",
"                        5 = Statistical,                              ",
"                                                                      ",
"                        6 = Attribute predicate,                      ",
"                                                                      ",
"                        7 = Mathematical) ==>                         ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"**********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"**********************************************************************"
},
}
```

```
,{
 4,                  /* number of input fields    */
 0,                  /* original cursor field     */
11,                  /* previous screen number    */
{                    /* length, row, col          */
{17,  7, 28},
{17, 10, 28},
{17, 10, 60},
{17, 13, 28}},
{                    /* screen text lines         */
"SDM ************** Member Attribute Derivation - Mapping ***********",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"   Mapping of attribute ==>                                          ",
"                                                                     ",
"                                                                     ",
"         onto attribute ==>                    of class ==>          ",
"                                                                     ",
"        using attribute ==>                                          ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"*********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"*********************************************************************"
},
}
```

```
,{
 3,                    /* number of input fields    */
 0,                    /* original cursor field     */
11,                    /* previous screen number    */
{                      /* length, row, col          */
{17,  7, 32},
{ 1, 10, 44},
{17, 13, 27}},
{                      /* screen text lines         */
"SDM ************ Member Attribute Derivation - Ordering ************",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"      Ordering of attribute ==>                                       ",
"                                                                      ",
"                                                                      ",
"      in (A = ascending, D = descending) ==>   order                   ",
"                                                                      ",
"                                                                      ",
"      within attribute ==>                                            ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"**********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"**********************************************************************"
},
}
```

```
,{
 2,                    /* number of input fields    */
 0,                    /* original cursor field     */
11,                    /* previous screen number    */
{                      /* length, row, col          */
{17,  8, 33},
{ 1, 11, 26}},
{                      /* screen text lines         */
"SDM ************ Member Attribute Derivation - Recursion ***********",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                On attribute ==>                                     ",
"                                                                     ",
"                                                                     ",
"                up to ==>   levels                                   ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"                                                                     ",
"*********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"*********************************************************************"
},
}
```

```
,{
 3,                  /* number of input fields    */
 0,                  /* original cursor field     */
11,                  /* previous screen number    */
{                    /* length, row, col          */
{ 1,  7, 60},
{17, 10, 29},
{17, 13, 29}},
{                    /* screen text lines         */
"SDM ************** Member Attribute Derivation - Set ***************",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"           (U = union, D = differenc, I = intersection) ==>         ",
"                                                                    ",
"                                                                    ",
"            of attribute ==>                                        ",
"                                                                    ",
"                                                                    ",
"           and attribute ==>                                        ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
 2,                  /* number of input fields    */
 0,                  /* original cursor field     */
11,                  /* previous screen number    */
{                    /* length, row, col          */
{ 3, 13, 54},
{17, 16, 43}},
{                    /* screen text lines         */
"SDM *********** Member Attribute Derivation - Statistical **********",
"                                                                    ",
"                                                                    ",
"                         (MIN = minimum,                            ",
"                                                                    ",
"                          MAX = maximum,                            ",
"                                                                    ",
"                          AVG = average,                            ",
"                                                                    ",
"                          SUM = sum,                                ",
"                                                                    ",
"                          TOT = total number,                       ",
"                                                                    ",
"                          UNQ = number of unique) ==>               ",
"                                                                    ",
"                                                                    ",
"                          of attribute ==>                          ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
11,                     /* number of input fields    */
 0,                     /* original cursor field     */
11,                     /* previous screen number    */
{                       /* length, row, col          */
{17,  5, 30},
{ 1,  7, 25},
{17,  7, 54},
{ 1,  9, 25},
{17,  9, 54},
{ 1, 11, 25},
{17, 11, 54},
{ 1, 13, 25},
{17, 13, 54},
{ 1, 15, 25},
{17, 15, 54}},
{                       /* screen text lines         */
"SDM ********** Member Attribute Derivation - Mathematical **********",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                attribute ==>                                       ",
"                                                                    ",
"        (+, -, *, /) ==>   attribute or 'literal' ==>               ",
"                                                                    ",
"        (+, -, *, /) ==>   attribute or 'literal' ==>               ",
"                                                                    ",
"        (+, -, *, /) ==>   attribute or 'literal' ==>               ",
"                                                                    ",
"        (+, -, *, /) ==>   attribute or 'literal' ==>               ",
"                                                                    ",
"        (+, -, *, /) ==>   attribute or 'literal' ==>               ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

```
,{
 3,                  /* number of input fields    */
 0,                  /* original cursor field     */
11,                  /* previous screen number    */
{                    /* length, row, col          */
{17,  4, 26},
{ 3, 15, 53},
{17, 17, 39}},
{                    /* screen text lines         */
"SDM ************ Member Attribute Derivation - Predicate ************",
"                                                                      ",
"                                                                      ",
"                                                                      ",
"            Attribute ==>                                             ",
"                                                                      ",
"                 (GT = greater then,                                  ",
"                  LT = less then,                                     ",
"                  EQ = equal to,                                      ",
"                  NE = not equal to,                                  ",
"                  GE = greater then or equal to,                      ",
"                  LE = less then or equal to,                         ",
"                  CT = contains,                                      ",
"                  PC = properly contains,                             ",
"                  IN = is contained in,                               ",
"                  PI = is properly contained in) ==>                  ",
"                                                                      ",
"            Attribute or 'literal' ==>                                ",
"                                                                      ",
"                                                                      ",
"**********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"**********************************************************************"
},
}
```

```
,{
 1,                  /* number of input fields    */
 0,                  /* original cursor field     */
 0,                  /* previous screen (n a)     */
{                    /* length, row, col          */
    {44,  9, 23}},
{                    /* screen text lines         */
"SDM ************************* VERIFICATION *************************",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"           Press ENTER to confirm the above action                  ",
"                                                                    ",
"           For     ==>                                              ",
"                                                                    ",
"           Press F10 to cancel the action.                          ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"                                                                    ",
"********************************************************************",
" F1=Help    F3=C Attr  F5=Insert   F7=Previous F9=Update  ESC=MS/DOS ",
" F2=Class   F4=M Attr  F6=Delete   F8=Next     F10=End    C/R=Enter  ",
"********************************************************************"
},
}
```

Appendix D

Error Messages

```
char msg[19][20] = {
            " invalid option",
            " file not found",
            " no update yet ",
            " I/O error     ",
            " required field",
            " invalid 1st c ",
            " invalid name  ",
            " non numeric   ",
            " F9 to update",
            "       DELETE      ",
            "        EDIT       ",
            "    GENERATE DD    ",
            "       PARSE       ",
            "  ACTION CANCELLED ",
};
char lmsg[19][80] = {
" The option selected for this field is not valid                    ",
" The file specified was not found, check full path name             ",
" The current component is not complete and cannot be updated yet ",
" An I/O error has occured while reading or writing your SDM       ",
" This field is required for the definition of this component      ",
" The first character must be alphabetic                           ",
" Valid characters are: alphanumeric and underscore                ",
" Only numeric values are valid in this field                      ",
" Component definition is complete, press F9 to save this definition"
};
```

Appendix E

System Code

```
#define BFSIZE 280
#define A1  0x46
#define FL  0xc6
#define RV  0x64
#define ENTER 13
#define ESC 20
#define OPEN  0
#define CLASS 1
#define CATR  2
#define MATR  3
#define HEADR 4
#define MSGL1 20
#define MSGL2 23
#define MSGC 00
#define DEL  10
#define EDIT 11
#define GEN  12
#define PRS  13
#define CAN  14
#define BADTYP 1
#define OPT   1
#define NOFIL 2
#define NOUPD 3
#define IOERR 4
#define REQ   5
#define CH1   6
#define CHR   7
#define NUM   8
#define UPD   9
#include <io.h>
#include <ctype.h>
#include <stdlib.h>
#include "scr.h"
#include "scr0.h"
#include "scr1.h"
#include "scr2.h"
#include "scr3.h"
#include "scr4.h"
#include "scr5.h"
#include "scr6.h"
#include "scr7.h"
#include "scr8.h"
#include "scr9.h"
#include "scr10.h"
#include "scr11.h"
#include "scr12.h"
#include "scr13.h"
```

```
#include "scr14.h"
#include "scr15.h"
#include "scr16.h"
#include "scr17.h"
#include "scr18.h"
#include "scr19.h"
};
#include "msg.h"
#include "rec.h"
#include "curses.h"
#include "scio.h"
char buf[300];
int  *sdmptr = &sdmu.hdr.fstclass;
FILE *sdm;
long int long0 = 0;
int recsize = sizeof(sdmu.rec);
int fstfree, curclass, fstclass, screen, lscreen, cursor, currec,
    fstcatr, fstmatr, rtyp, numfree, numrec, filestat, oset, c, upd;
char fp[44];
char opt[1];
main() {
   initscr();
   cbreak();
   noecho();
   refresh();
   clear();
   mnu();
   clear();
   move(0,0);
   refresh();
   endwin();
   exit();
}
mnu() {
   int i, j, e = 0;
   opt[0] = 0;
   while(1) {
      opt[0] = toupper(opt[0]);
      if (opt[0]) switch (opt[0]) {
         case 'D': e = delsdm(); opt[0] = 0; break;
         case 'E': e = edit();   opt[0] = 0; break;
         case 'G': e = gen();    opt[0] = 0; break;
         case 'P': e = prs();    opt[0] = 0; break;
         case 'X': return;
         default : e = 1;
      }
      scio(0,0,opt,e);
      e = 0;
   }
}
```

```
id(a) int a;{                    /* get sdm full path name */
   int i, j, k, m, e=0;
   char buf[43];
   for (i = 0; i < sizeof(buf);i++) { buf[i] = 0; fp[i] = 0;}
   filestat = 2;
   while(filestat > 1) {
      scio(1,0,buf,e);
      for (i=k=m=0;i<scr[1].fields;i++) {
         for(j=0;j<scr[1].floc[i][0]&&buf[k+j]>' ';)
            fp[m++] = buf[k+j++];
         if (i == 0) fp[m++] = ':';
         k += scr[1].floc[i][0];
         if (i < scr[1].fields-1 && buf[k]>' ') fp[m++] = '/';
      }
      fp[m++] = '.'; fp[m++] = 's'; fp[m++] = 'd'; fp[m++] = 'm';
      if (sdm = fopen(fp,"r+")) filestat = 0;
      else if (sdm = fopen(fp,"w+")) filestat = 1; else filestat = 2;
      e = filestat;
   }
   return(scio(19,0,fp,a));
}
delsdm() {                              /* delete an sdm file */
   if (id(DEL) == 68) return(CAN);      /* verify action      */
   if (filestat) return(NOFIL);         /* good sdm name?     */
   if (c = unlink(fp)) return(c);       /* unlink             */
   return(OK);
}
gen() {                                 /* generate a data    */
   if (id(GEN) == 68) return(CAN);      /* dictionary         */
   if (filestat) return(NOFIL);
   return(OK);                          /* not implemented    */
}
prs() {                                 /* parse an sdm       */
   if (id(PRS) == 68) return(CAN);      /* definition         */
   if (filestat) return(NOFIL);
   return(OK);                          /* not implemented    */
}
edit() {                                /* edit an sdm def    */
   int i, rc, e = 0;
   setbuf(sdm,NULL);
   if (id(EDIT) == 68) return(CAN);     /* verify action      */
   if (filestat == 0) {                 /* open old sdm       */
      fread(sdmptr,recsize,1,sdm);
      curclass = currec = fstclass = sdmu.hdr.fstclass;
      fstfree = sdmu.hdr.fstopen;
      numrec = sdmu.hdr.nrec;
      numfree = sdmu.hdr.nopen;
      getrec(currec);
   }
```

```
else {                                  /* open new sdm       */
   numrec = currec = curclass = fstclass = 1;
   fstfree = numfree = fstcatr = fstmatr = 0;
   sdmu.rec.pre    = 0;
   sdmu.rec.nxt    = 0;
   sdmu.rec.catr   = 0;
   sdmu.rec.matr   = 0;
   sdmu.rec.typ    = CLASS;
}
rtyp = CLASS;
screen = 2;
oset = upd = 0;
while (screen > 0)                  /* if 0 return to main menu */
   switch(scio(screen,cursor,sdmu.rec.buf+oset,e)) {
   case OK: lscreen = screen;
            switch(screen){          /* validate screen input */
      case  2: e = proc2(sdmu.rec.buf+oset); break;
      case  3: e = proc3(sdmu.rec.buf+oset); break;
      case  4: e = proc4(sdmu.rec.buf+oset); break;
      case  5: e = proc5(sdmu.rec.buf+oset); break;
      case  6: e = proc6(sdmu.rec.buf+oset); break;
      case  7: e = proc7(sdmu.rec.buf+oset); break;
      case  8: e = proc8(sdmu.rec.buf+oset); break;
      case  9: e = proc9(sdmu.rec.buf+oset); break;
      case 10: e = proc10(sdmu.rec.buf+oset); break;
      case 11: e = proc11(sdmu.rec.buf+oset); break;
      case 12: e = proc12(sdmu.rec.buf+oset); break;
      case 13: e = proc13(sdmu.rec.buf+oset); break;
      case 14: e = proc14(sdmu.rec.buf+oset); break;
      case 15: e = proc15(sdmu.rec.buf+oset); break;
      case 16: e = proc16(sdmu.rec.buf+oset); break;
      case 17: e = proc17(sdmu.rec.buf+oset); break;
      case 18: e = proc18(sdmu.rec.buf+oset); break;
   } break;
   case 59: hlp(e); break;           /* process function keys */
   case 60: e = cls(); break;
   case 61: e = ctr(); break;
   case 62: e = mtr(); break;
   case 63: e = add(); break;
   case 64: e = del(); break;
   case 65: e = pre(); break;
   case 66: e = nxt(); break;
   case 67: e = updt(); break;
   case 68: screen = scr[screen].prevscr;
            oset -= soffset(screen);
            cursor = scr[screen].cursor;
            e = OK;
}
```

```
    sdmu.hdr.fstclass = fstclass;        /* update header and */
    sdmu.hdr.fstopen  = fstfree;         /* close file        */
    sdmu.hdr.nrec     = numrec;
    sdmu.hdr.nopen    = numfree;
    sdmu.hdr.typ      = HEADR;
    for (i=0;i<BFSIZE;i++) sdmu.hdr.buf[i] = 0;
    fseek(sdm,long0,0);
    fwrite(sdmptr,recsize,1,sdm);
    fclose(sdm);
    return(OK);
}
proc2(b) char b[];{
    int i;
    cursor = 0;
    if (c = nblank(b,17)) return(c);            /* required field */
    if (c = goodname(b)) return(c);             /* invalid name   */
    i = foffset(screen,1);
    b[i] = toupper(b[i]);
    switch(b[i]) {
          case 'B': screen = 3; break;
          case 'S': screen = 4; break;
          case 'G': screen = 8; break;
          default: cursor = 1; return(OPT);    /* invalid option */
    }
    oset += soffset(lscreen);
    cursor = scr[screen].cursor;
    return(OK);
}
proc3(b) char b[];{
    int i;
    b[0] = toupper(b[0]);
    switch(b[0]) {
          case 'Y':
          case 'N': break;
          default: cursor = 0; return(OPT);     /* invalid option */
    }
    for (cursor = 1; cursor < 11;cursor++) {
          i = foffset(screen,cursor);
          if (nblank(b+i,17)==OK)               /* if non blank   */
             if (c = goodname(b+i)) return(c);  /* invalid name   */
    }
    upd = 1;                                    /* allow update   */
    cursor = 0;
    return(UPD);
}
proc4(b) char b[];{
    int i;
    cursor = 0;
    if (c = nblank(b,17)) return(c);            /* required field */
    else if (c = goodname(b)) return(c);        /* invalid name   */
    i = foffset(screen,1);
```

```
    switch(b[i]) {
          case '1': return(UPD);
          case '2': screen = 5; break;
          case '3': screen = 6; break;
          case '4': screen = 7; break;
          default: cursor = 1; return(OPT);    /* invalid option */
    }
    oset += soffset(lscreen);
    cursor = scr[screen].cursor;
    return(OK);
}
proc5(b) char b[];{
    int i;
    b[0] = toupper(b[0]);
    switch(b[0]) {
          case 'U':
          case 'D':
          case 'I': break;
          default: cursor = 0; return(OPT);     /* invalid option */
    }
    cursor = 1;
    if (c = nblank(b+1,17)) return(c);          /* required field */
    if (c = goodname(b+1)) return(c);           /* invalid name   */
    cursor = 2;
    if (c = nblank(b+18,17)) return(c);         /* required field */
    if (c = goodname(b+18)) return(c);          /* invalid name   */
    cursor = 0;
    upd = 1;                                    /* allow update   */
    return(UPD);
}
proc6(b) char b[];{
    int i;
    cursor = 0;
    if (c = nblank(b,2)) return(c);             /* required field */
    if (c = numb(b,2)) return(c);               /* non numeric    */
    cursor = 1;
    if (c = nblank(b+2,2)) return(c);           /* required field */
    if (c = numb(b+2,2)) return(c);             /* non numeric    */
    upd = 1;                                    /* allow update   */
    cursor = 0;
    return(UPD);
}
proc7(b) char b[];{
    int i, j;
    cursor = 0;
    if (c = nblank(b,17)) return(c);            /* required field */
    if (c = goodname(b)) return(c);             /* invalid name   */
    cursor = 1;
    i = foffset(screen,cursor);
    for (j = 0; j < 2; j++)                     /* convert to cap */
       b[i+j] = toupper(b[i+j]);                /* & check value  */
```

```
   if (!((strncmp(b+i,"GT",2) == 0 ) ||
         (strncmp(b+i,"LT",2) == 0 ) ||
         (strncmp(b+i,"GE",2) == 0 ) ||
         (strncmp(b+i,"LE",2) == 0 ) ||
         (strncmp(b+i,"EQ",2) == 0 ) ||
         (strncmp(b+i,"NE",2) == 0 ) ||
         (strncmp(b+i,"CT",2) == 0 ) ||
         (strncmp(b+i,"PC",2) == 0 ) ||
         (strncmp(b+i,"IN",2) == 0 ) ||
         (strncmp(b+i,"PI",2) == 0 )))
      return(OPT);                              /* invalid option */
   cursor = 2;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);           /* required field */
   upd = 1;                                     /* allow update   */
   cursor = 0;
   return(UPD);
}
proc8(b) char b[];{
   int i, f = 0;
   cursor = 0;
   if (c = nblank(b,17)) return(c);             /* required field */
   if (c = goodname(b)) return(c);              /* invalid name   */
   cursor = 1;
   i = foffset(screen,cursor);
   if (nblank(b+i,17)==OK) {                    /* if non blank   */
      if (c = goodname(b+i)) return(c);         /* invalid name   */
      f = 1;
   }
   for (cursor = 2; cursor < 12 ;cursor++) {
      i = foffset(screen,cursor);
      if (nblank(b+i,17)==OK) {                 /* if non blank   */
         if (c = goodname(b+i)) return(c);      /* invalid name   */
         f = 1;
      }
   }
   cursor = 1;
   if (!f) return(REQ);                         /* required field */
   upd = 1;                                     /* allow update   */
   cursor = 0;
   return(UPD);
}
proc9(b) char b[];{
   int i, j;
   cursor = 0;
   if (c = nblank(b,17)) return(c);             /* required field */
   if (c = goodname(b)) return(c);              /* invalid name   */
   cursor = 1;
   i = foffset(screen,cursor);
   if (! nblank(b+i,17))                        /* if nonblank    */
      if (c = goodname(b+1)) return(c);         /* invalid name   */
```

```
    for (i = 3; i < 5; i++) {
       j = foffset(screen,i);
       b[j] = toupper(b[j]);
       switch(b[j]) {
             case 'Y':
             case 'N': break;
             default: cursor = i; return(OPT);  /* invalid option */
       }
    }
    cursor = 5;
    i = foffset(screen,5);
    if (c = nblank(b+i,2)) return(c);           /* required field */
    if (c = numb(b+i,2)) return(c);             /* non numeric    */
    cursor = 6;
    i = foffset(screen,6);
    if (c = nblank(b+i,2)) return(c);           /* required field */
    if (c = numb(b+i,2)) return(c);             /* non numeric    */
    cursor = 7;
    i = foffset(screen,7);
    if (! nblank(b+i,3)) {                      /* if nonblank    */
       for (j = 0; j < 3; j++)                  /* then check     */
          b[i+j] = toupper(b[i+j]);             /* value          */
       if (!((strncmp(b+i,"MIN",3) == 0 ) ||
             (strncmp(b+i,"MAX",3) == 0 ) ||
             (strncmp(b+i,"AVG",3) == 0 ) ||
             (strncmp(b+i,"SUM",3) == 0 ) ||
             (strncmp(b+i,"TOT",3) == 0 ) ||
             (strncmp(b+i,"UNQ",3) == 0 )))
          return(OPT);                          /* invalid option */
       cursor = 8;
       i = foffset(screen,8);
       if (c = nblank(b+i,17)) return(c);       /* required field */
       if (c = goodname(b+i)) return(c);        /* invalid name   */
    }
    upd = 1;                                    /* allow update  */
    cursor = 0;
    return(UPD);
}
proc10(b) char b[];{
    int i;
    cursor = 0;
    if (c = nblank(b,17)) return(c);            /* required field */
    if (c = goodname(b)) return(c);             /* invalid name   */
    cursor = 1;
    i = foffset(screen,cursor);
    if (nblank(b+i,17) == OK)                   /* if nonblank    */
       if (c = goodname(b+1)) return(c);        /* invalid name   */
    cursor = 3;
    i = foffset(screen,cursor);
```

```
    if (nblank(b+i,17) == OK) {                 /* if nonblank    */
       if (c = goodname(b+i)) return(c);        /* invalid name   */
       cursor = 4;
       i = foffset(screen,cursor);
       if (c = nblank(b+i,17)) return(c);       /* required field */
       if (c = goodname(b+i)) return(c);        /* invalid name   */
    }
    for (cursor = 5; cursor < 9; cursor++) {
       i = foffset(screen,cursor);
       b[i] = toupper(b[i]);
       switch(b[i]) {
             case 'Y':
             case 'N': break;
             default:  return(OPT);             /* invalid option */
       }
    }
    cursor = 9;
    i = foffset(screen,cursor);
    if (c = nblank(b+i,2)) return(c);           /* required field */
    if (c = numb(b+i,2)) return(c);             /* non numeric    */
    cursor = 10;
    i = foffset(screen,cursor);
    if (c = nblank(b+i,2)) return(c);              /* required field */
    if (c = numb(b+i,2)) return(c);                /* non numeric    */
    cursor = 11;
    i = foffset(screen,cursor);
    b[i] = toupper(b[i]);
    switch(b[i]) {
          case 'Y': screen = 11;
                    oset += soffset(lscreen);
                    cursor = scr[screen].cursor;
                    return(OK);
          case 'N': cursor = 12;
                    i = foffset(screen,cursor);
                    if (c = nblank(b+i,17)) return(c);
                    if (c = goodname(b+i)) return(c);
                    cursor = 13;
                    i = foffset(screen,cursor);
                    if (c = nblank(b+i,17)) return(c);
                    if (c = goodname(b+i)) return(c);
                    cursor = 14;
                    i = foffset(screen,cursor);
                    if (c = nblank(b+i,17)) return(c);
                    if (c = goodname(b+i)) return(c);
                    break;
          default:  return(OPT);                /* invalid option */
    }
    upd = 1;                                    /* allow update   */
    cursor = 0;
    return(UPD);
}
```

```
proc11(b) char b[];{
   switch(b[0]) {
         case '1': screen = 12; break;
         case '2': screen = 13; break;
         case '3': screen = 14; break;
         case '4': screen = 15; break;
         case '5': screen = 16; break;
         case '6': screen = 18; break;
         case '7': screen = 17; break;
         default: return(OPT);                    /* invalid option */
   }
   oset += soffset(lscreen);
   cursor = scr[screen].cursor;
   return(OK);
}
proc12(b) char b[];{
   int i;
   for (cursor = 0; cursor < 4;cursor++) {
      i = foffset(screen,cursor);
      if (c = nblank(b+i,17)) return(c);          /* required field */
      if (c = goodname(b+i)) return(c);           /* invalid name   */
   }
   upd = 1;                                       /* allow update   */
   cursor = 0;
   return(UPD);
}
proc13(b) char b[];{
   int i;
   cursor = 0;
   if (c = nblank(b,17)) return(c);               /* required field */
   if (c = goodname(b)) return(c);                /* invalid name   */
   cursor = 1;
   i = foffset(screen,cursor);
   b[i] = toupper(b[i]);
   switch(b[i]) {
         case 'A':
         case 'D': break;
         default:  return(OPT);                   /* invalid option */
   }
   cursor = 2;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);             /* required field */
   if (c = goodname(b+i)) return(c);              /* invalid name   */
   upd = 1;                                       /* allow update   */
   cursor = 0;
   return(UPD);
}
proc14(b) char b[];{
   int i;
   cursor = 0;
   if (c = nblank(b,17)) return(c);               /* required field */
```

```
   if (c = goodname(b)) return(c);              /* invalid name   */
   cursor = 1;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,1)) return(c);            /* required field */
   if (c = numb(b+i,1)) return(c);              /* non numeric    */
   upd = 1;                                     /* allow update   */
   cursor = 0;
   return(UPD);
}
proc15(b) char b[];{
   int i;
   cursor = 0;
   b[0] = toupper(b[0]);
   switch(b[0]) {
         case 'U':
         case 'D':
         case 'I': break;
         default: return(OPT);                  /* invalid option */
   }
   cursor = 1;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);           /* required field */
   if (c = goodname(b+i)) return(c);            /* invalid name   */
   cursor = 2;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);           /* required field */
   if (c = goodname(b+i)) return(c);            /* invalid name   */
   upd = 1;                                     /* allow update   */
   cursor = 0;
   return(UPD);
}
proc16(b) char b[];{
   int i;
   cursor = 0;
   for (i = 0; i < 3; i++)
      b[i] = toupper(b[i]);
   if (!((strncmp(b,"MIN",3) == 0 ) ||
         (strncmp(b,"MAX",3) == 0 ) ||
         (strncmp(b,"AVG",3) == 0 ) ||
         (strncmp(b,"SUM",3) == 0 ) ||
         (strncmp(b,"TOT",3) == 0 ) ||
         (strncmp(b,"UNQ",3) == 0 )))
      return(OPT);                              /* invalid option */
   cursor = 1;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);           /* required field */
   if (c = numb(b+i)) return(c);                /* invalid name   */
   upd = 1;                                     /* allow update   */
   cursor = 0;
   return(UPD);
}
```

```
proc17(b) char b[];{
   int i, f = 0;
   cursor = 0;
   if (c = nblank(b,17)) return(c);              /* required field */
   if (c = goodname(b)) return(c);               /* invalid name   */
   for (cursor = 1; cursor < 10; cursor +=2) {
      i = foffset(screen,cursor);
      switch(b[i]) {
            case '+':
            case '-':
            case '*':
            case '/': f = 1;
                      i = foffset(screen,cursor+1);
                      if (c = nblank(b+i,17)) return(c);
                      if (c = goodname(b+i)) return(c);
            case ' ': break;
            default : return(OPT);
      }
   }
   if (!f) return(REQ);                           /* required field */
   upd = 1;                                       /* allow update   */
   cursor = 0;
   return(UPD);
}
proc18(b) char b[];{
   int i, j;
   cursor = 0;
   if (c = nblank(b,17)) return(c);              /* required field */
   if (c = goodname(b)) return(c);               /* invalid name   */
   cursor = 1;
   i = foffset(screen,cursor);
   for (j = 0; j < 2; j++)                       /* convert to cap */
      b[i+j] = toupper(b[i+j]);                  /* & check value  */
   if (!(strncmp(b+i,"GT",2) == 0 ) ||
         (strncmp(b+i,"LT",2) == 0 ) ||
         (strncmp(b+i,"GE",2) == 0 ) ||
         (strncmp(b+i,"LE",2) == 0 ) ||
         (strncmp(b+i,"EQ",2) == 0 ) ||
         (strncmp(b+i,"NE",2) == 0 ) ||
         (strncmp(b+i,"CT",2) == 0 ) ||
         (strncmp(b+i,"PC",2) == 0 ) ||
         (strncmp(b+i,"IN",2) == 0 ) ||
         (strncmp(b+i,"PI",2) == 0 )))
      return(OPT);                               /* invalid option */
   cursor = 2;
   i = foffset(screen,cursor);
   if (c = nblank(b+i,17)) return(c);            /* required field */
   upd = 1;                                      /* allow update   */
   cursor = 0;
   return(UPD);
}
```

```
nblank(b,n) int n; char b[]; {
   int i;
   for (i = 0;i < n;i++) switch (b[i]) {
      case  0 :
      case ' ': break;
      default : return(OK);                        /* non blank field   */
   }
   return(REQ);                                    /* required field    */
}
numb(b,n) char b[]; int n;{
   int i, f = 0;
   for (i=0;i<n;i++) switch(b[i]) {
      case '0':
      case '1':
      case '2':
      case '3':
      case '4':
      case '5':
      case '6':
      case '7':
      case '8':
      case '9': f = 1; break;
      case  0 :
      case ' ': if (f) {
                  b[i] = 0;
                  while (++i < n) {
                     if (b[i] == ' ' ) b[i] = 0;
                     else if (b[i])
                              return(NUM);         /* imbedded blank    */
                  }
                } break;
      default : return(NUM);                       /* non numeric       */
   }
   return(OK);
}
goodname(b) char b[]; {
   int i;
   if (b[0] >= 'a' && b[0] <= 'z')                 /* convert 1st char  */
       b[0] = b[0] + 'A' - 'a';                    /* to upper case     */
   else if (b[0] < 'A' || b[0] > 'Z')
            return(CH1);                           /* invalid 1st char  */
   for (i = 1;i < 17;i++) {                        /* valid characters  */
      if (!((b[i] >= 'a' && b[i] <= 'z') ||        /* lower case alpha  */
            (b[i] >= 'a' && b[i] <= 'z') ||        /* upper case alpha  */
            (b[i] >= '0' && b[i] <= '9') ||        /* numeric           */
             b[i] == ' ' || b[i] == 0    ||        /* blank or null     */
             b[i] == '_')) return(CHR);            /* underscore        */
```

```
      if (b[i] == ' ' || b[i] == 0) {
         b[i] = 0;
         while (++i < 17) {
            if (b[i] == ' ' ) b[i] = 0;
            else if (b[i]) return(CHR);         /* imbedded blank    */
         }
      }
   }
   return(OK);
}
hlp(e) int e; {
   int hscreen, w[4];
   if (e) {
      w[0] = w[3] = MSGL1;
      w[1] = MSGC;
      w[2] = MSGC + 76;
      wattron(w,RV);
      mvaddstr(MSGL1,MSGC,lmsg[e-1]);
      w[0] = w[3] = MSGL2;
      w[1] = MSGC;
      w[2] = MSGC + 24;
      wattron(w,RV);
      mvaddstr(MSGL2,MSGC,"PRESS ANY KEY TO CONTINUE");
      getch();
   }
/* else {                              */     /* tutorial             */
/*    hscreen = screens - screen;      */     /* not implemented      */
/*    scio(hscreen,0,b,0);             */
/* }                                   */
   return(OK);
}
cls(){                                         /* display current class */
   screen = 2;
   cursor = 0;
   oset = upd = 0;
   rtyp = CLASS;
   if (currec==curclass) return(OK);
   return(getrec(curclass));
}
ctr(){                                         /* display first class   */
   screen =  9;                                /* attribute of current  */
   oset = upd = cursor = 0;                    /* class                 */
   rtyp = CATR;
   if (currec==fstcatr) return(OK);
   if (fstcatr) return(getrec(fstcatr));
   return(add());
}
```

```
mtr(){                                          /* display first member  */
   screen = 10;                                 /* attribute of current  */
   oset = upd = cursor = 0;                     /* class                 */
   rtyp = MATR;
   if (currec==fstmatr) return(OK);
   if (fstmatr) return(getrec(fstmatr));
   return(add());
}
pre(){
   if (sdmu.rec.pre) {                          /* display previous    */
      if (c = getrec(sdmu.rec.pre))             /* record of this type */
         return(c);
      if (sdmu.rec.typ == CLASS) curclass = currec;
   }
   oset = upd = 0;
   switch (rtyp) {
      case CLASS: screen =  2; break;
      case CATR : screen =  9; break;
      case MATR : screen = 10; break;
      default : return(BADTYP);
   }
   cursor = scr[screen].cursor;
   return(OK);
}
nxt(){                                          /* display next record */
   if (sdmu.rec.nxt)                            /* of this type        */
      if (c = getrec(sdmu.rec.nxt)) return(c);
   oset = upd = 0;
   switch (rtyp) {
      case CLASS: screen =  2; break;
      case CATR : screen =  9; break;
      case MATR : screen = 10; break;
      default : return(BADTYP);
   }
   cursor = scr[screen].cursor;
   return(OK);
}
updt() {
   if (upd) {
      oset = upd = 0;
      switch (rtyp) {
         case CLASS: screen =  2; break;
         case CATR : screen =  9; break;
         case MATR : screen = 10; break;
         default : return(BADTYP);
      }
      cursor = scr[screen].cursor;
      return(putrec(currec));
   }
   else return(NOUPD);
}
```

```
del(){                                  /* delete current record from */
   int scatr = sdmu.rec.catr,           /* sdm file and update linked */
       smatr = sdmu.rec.matr,           /* list of open records       */
       sprev = sdmu.rec.pre,
       snext = sdmu.rec.nxt,
       stype = sdmu.rec.typ,
       scur  = currec,
       ncur  = 0,
       temp;
   char b[45];
   for (temp = 0;temp < 45; temp++) b[temp] = 0;
   for (temp = 0;temp < 17; temp++)
      b[temp] = sdmu.rec.buf[temp];     /* get name of record     */
   if (scio(19,0,b,DEL)==68) return(OK);  /* and verify delete      */
   if (snext) {
       ncur = snext;                                /* update pointer  */
      if (c = getrec(snext)) return(c);             /* in next record  */
      sdmu.rec.pre = sprev;
      if (c = putrec(snext)) return(c);
   }                                                /* and prev record */
   if (sprev) {
      ncur = sprev;
      if (c = getrec(sprev)) return(c);
      sdmu.rec.nxt = snext;
      if (c = putrec(sprev)) return(c);
   }
   else {                                           /* if no prev rec  */
      if (stype == CLASS)                           /* and type CLASS  */
         fstclass = curclass = snext;               /* update fstclass */
      else {                                        /* else update the */
         if (c = getrec(curclass)) return(c);       /* current class   */
         if (rtyp == CATR)                          /* if type CATR    */
             sdmu.rec.catr = snext;                 /* update  catr    */
          else sdmu.rec.matr = snext;               /* else update matr*/
         if (c = putrec(currec)) return(c);
      }
      if (!ncur) ncur = curclass;
   }
   pushopen(scur);
   if (stype == CLASS) {                    /* if current record is */
      if (scatr) {                          /* class, then all of   */
         if (c == getrec(scatr)) return(c);  /* its class attributes */
         while (sdmu.rec.nxt) {              /* must also be deleted */
            temp = currec;
            if (c == getrec(sdmu.rec.nxt)) return(c);
            pushopen(temp);
         }
      }
```

```
      if (smatr) {                              /* and also all of its  */
         if (c == getrec(smatr)) return(c);     /* member attributes    */
         while (sdmu.rec.nxt) {
            temp = currec;
            if (c == getrec(sdmu.rec.nxt)) return(c);
            pushopen(temp);
         }
      }
   }
   if (c == getrec(ncur)) return(c);
   oset = upd = 0;
   switch (sdmu.rec.typ) {
      case CLASS: screen =  2; break;
      case CATR : screen =  9; break;
      case MATR : screen = 10; break;
      default : return(BADTYP);
   }
   cursor = scr[screen].cursor;
   return(OK);
}
pushopen(n) int n;{
   int i;                                       /* add a record to the  */
   if (n<1) return(ERR );                       /* linked list of open  */
   sdmu.rec.nxt = fstfree;                      /* records              */
   sdmu.rec.catr = sdmu.rec.matr = sdmu.rec.pre = 0;
   sdmu.rec.typ = OPEN;
   for (i=0;i<BFSIZE;i++) sdmu.hdr.buf[i] = 0;
   if (c = putrec(n)) return(c);
   fstfree = n;
   return(OK);
}
add() {                                         /* get a record from    */
   int snext = sdmu.rec.nxt,                    /* the linked list of   */
       stype = sdmu.rec.typ,                    /* open records if any  */
       sfree = fstfree,                         /* are free, or add one */
       scur  = currec,                          /* to the end of file   */
       i;
   if (fstfree) currec = fstfree;               /* update pointers to   */
   else currec = ++numrec;                      /* new rec in pre & nxt */
   if (rtyp == stype) {                         /* if type unchanged    */
      sdmu.rec.nxt = currec;                    /* then update the last */
      if (c = putrec(scur)) return(c);          /* recs next pointer    */
      if (snext){                               /* if there is a next   */
         i  = currec;
         if (c = getrec(snext)) return(c);      /* read it and update   */
         currec = i;
         sdmu.rec.pre = currec;                 /* its previous pointer */
         if (c = putrec(snext)) return(c);
      }
      if (rtyp == CLASS) curclass = currec;
   }
```

```
   else {                                    /* if type has changed  */
      if (scur != curclass) {                /* then class rec must  */
         scur = currec;                      /* be updated           */
         if (c = getrec(curclass)) return(c);
         currec = scur;
      }
      if (rtyp == CATR)
         sdmu.rec.catr = fstcatr = currec;
      else
         sdmu.rec.matr = fstmatr = currec;
      if (c = putrec(curclass)) return(c);
      snext = scur = 0;                      /* rec of new typ has   */
   }                                         /* no pre or nxt        */
   if (sfree) {
      if (c = getrec(sfree)) return(c);
      fstfree = sdmu.rec.nxt;
   }
   oset = upd = 0;
   switch (rtyp) {
      case CLASS: screen =  2; break;
      case CATR : screen =  9; break;
      case MATR : screen = 10; break;
      default : return(BADTYP);
   }
   cursor = scr[screen].cursor;
   for (i=0;i<BFSIZE;i++) sdmu.rec.buf[i] = 0;
   sdmu.rec.catr = 0;
   sdmu.rec.matr = 0;
   sdmu.rec.pre  = scur;
   sdmu.rec.nxt  = snext;
   sdmu.rec.typ  = rtyp;
   if (c = putrec(currec)) return(c);
   return(OK);
}
getrec(n) int n; {                           /* read a specific record */
   long int i = n;
   if (fseek(sdm,i*recsize,0)) return(3);
   if (fread(sdmptr,recsize,1,sdm)!=1) return(IOERR);
   currec = n;
   if (sdmu.rec.typ == CLASS) {
      curclass = currec;
      fstcatr = sdmu.rec.catr;
      fstmatr = sdmu.rec.matr;
   }
   return(OK);
}
```

```
putrec(n) int n; {                          /* write a specific record */
   long int i = n;
   if (fseek(sdm,i*recsize,0)) return(3);
   if (fwrite(sdmptr,recsize,1,sdm)!=1) return(IOERR);
   return(OK);
}
foffset(s, f) int s, f; {                   /* compute the offset of a */
   int i, k;                                /* given input field       */
   for (i = k = 0;i < f;i++)
      k += scr[s].floc[i][0];
   return(k);
}
soffset(s) int s; {                         /* compute the total offset */
   int i, k;                                /* for input on one screen  */
   for (i = k = 0;i < scr[s].fields;i++)
      k += scr[s].floc[i][0];
   return(k);
}
esc(){
   int i;
   sdmu.hdr.fstclass = fstclass;
   sdmu.hdr.fstopen  = fstfree;
   sdmu.hdr.nrec     = numrec;
   sdmu.hdr.nopen    = numfree;
   sdmu.hdr.typ      = HEADR;
   for (i=0;i<BFSIZE;i++) sdmu.hdr.buf[i] = 0;
   fseek(sdm,long0,0);
   fwrite(sdmptr,recsize,1,sdm);            /* update header record  */
   fclose(sdm);                             /* and close file        */
   clear();
   move(0,0);
   refresh();
   endwin();
   exit();
}
```

References

1) Andriole, S. "Interactive Computer-Based Systems: Design & Developement", Petrocelli Books, Princeton, NJ, 1983

2) Gains B. The technology of interaction - dialogue programming rules", International Journal of Man-Machine Studies, 14, 1981

3) Hammer, M. and D. McLeod. "The semantic data model: A modeling Mechanism for data base applications", SIGMOD (ACM) Int. Conf. on Management of Data, Austin, TX, May 31, June 1-2, 1978

4) Hammer, M. and D. McLeod. "The semantic data model: A conceptual data modeling mechanism", Advances in data base management, Ed. T. Rullo, Heyden, Philadelphia, PN, 1980

5) Hammer, M. and D. McLeod. "Database description with SDM: A Semantic Database Model", ACM Trans Database Syst, 6:3, 1981

6) Miller, G. "The magical number seven, plus or minus two: Some limits on our capacity for processing information", Psycological Review, 63:2, 1956

7) Monk A. "Fundamentals of Human-Computer Interaction", Academic Press, New York, NY, 1984

8) Lane, R. "Senantic Database Model Language (SDML): Grammer Specification and Parser", Masters Thesis, Kansas State University, 1986

AN INTERACTIVE SYSTEM FOR THE DEFINITION OF A SEMANTIC DATA MODEL

by

GREGORY DALE WOOD

B.S., Kansas State University, 1974

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1987

The Semantic Data Model (SDM) was developed for the conceptual modeling of application environments. An interactive system for SDM definition will enable database designers to more easily create these models, with standard results.

The system will allow the user to enter and maintain one or more SDM definitions, delete an entire SDM definition, parse the definition of an SDM for syntactic correctness, and to generate a static data dictionary from an SDM definition.

These functions are accomplished through menu selection and fill in the blank, formatted screens, which provide initial guidance in SDM definition. Futher guidance is available through error messages and expanded descriptions thereof, as well as general help screens about the system and related SDM syntax.